DOI:
10.1109/SAI.2016.7556114

Document Version
Peer reviewed version

Statistical Approach towards Malware Classification and Detection

Vida Ghanaei, Costas S. Iliopoulos, Richard E. Overill
Department of Informatics, King's College London
Strand, London, WC2R 2LS, UK
{vida.ghanaei, c.iliopoulos, richard.overill}@kcl.ac.uk

***Abstract*—Anti-virus companies receive extensive quantities of malware variants daily; therefore it is essential to automatically classify them into their corresponding malware family. Here, we apply an effective statistical approach to identify, and to render critical malicious patterns into malware families, which are essential elements of automated classification of known and unknown malware variants in large quantities. Critical malicious patterns are the most frequent basic blocks, which are present most often in one specific malware family, and comparatively less in all other malware families. By computing the distribution frequency of each distinct basic block residing in all the malware families, the importance of being a potential representative of a critical malicious pattern for a specific malware family is measured. This value is carefully computed by considering the population of each malware family, and the distribution frequency ratio of every distinct basic block among the different malware families. The results show that known and unknown malware variants can be effectively and accurately classified into their related malware family using this approach.**

***Keywords*—Malware Classification; Statistical; Pattern Matching; Malicious Features; Shared Code;**

I. Introduction

Malware is considered to be a major computer security problem as it can attach to other computer programs and infect them. Infection is defined as unwanted modification of other programs to include a (possibly evolved) version of itself [6]. In recent years, malware has become more sophisticated, goal oriented, and possibly developed by a group of authors rather than a single author. It is applied to warfare between governments and organisations with the aim of accessing sensitive information and affecting each others' reputations. Furthermore, the flow of malicious software variants arriving at, and gaining access into, our networks and computer systems is so enormous that it is impossible to handle them individually. Also, malware replication speed urges the need for automated analysis and classification as it is impossible to analyse and detect otherwise. Based on reports published by different antivirus research laboratories, huge amounts of known and unknown malware variants reach their honeypots, which cannot be analysed manually. In this paper, we suggest that every approach towards malware analysis, detection, classification, and clustering should be either automated or designed in such a way that makes future automation possible, unless it is intended to analyse only a limited number of malware variants for research purposes. On the other hand, modern malware employs many different obfuscation techniques to hide itself from antivirus scanners; therefore it is no longer possible to identify later mutations of malware variants, which belong to the same malware family, by signature based anti-virus scanners, as its structure and code is changed.

Although many solutions has been proposed to tackle this problem, still malware authors are the winners of the race and every day infections of more systems demonstrates the limitations of the current detection solutions. Different studies are published on malware analysis and classification, which look into different features of malicious software. However, many of the previous works follow the same approach of antivirus scanners, in terms of generating a set of signatures to recognise malware, such as works presented in [11], [18], [19], [21]. Signature-based malware detection approaches are limited to the predefined signatures in the database, and the unknown malware variants are not identifiable. Also, these approaches are designed in a way, which makes them dependent on limited dataset, and therefore, the extracted features are also limited to the state of the training dataset. This limitation highly effects the future malware variants detection, as every time a malware replicates it carries some new code. Thus, over the time, and after few generations, malware variants from the same malware family, become unrecognisable from its original form, and difficult to identify if to be analysed exclusively. Authors in another study [16], has applied exact similarity measurement technique, which limit the detection to exact matches, and dependent to the training dataset. Some of the studies use virtualisation or emulation techniques to observe malicious behaviour, such as the works presented by authors in [8], [12], [21], which malware authors bypass them by embedding anti-virtualisation, anti-emulation techniques in the malicious code.

In this paper, a supervised learning method is applied to classify new malware variants into their related malware families based on an effective statistical approach to identify, and to render critical malicious patterns into malware families [20]. To identify the most critical malicious patterns in every malware family, the authors have relied on the shared basic blocks among different malware variants, which potentially occur in "one" specific family at most. However, the shared basic blocks in-between different malware families, are not interesting for classification purposes, as certain functions are in-common amongst all the malicious software variants, and can even be in-common with non-malicious software. To achieve this goal, the frequency distribution ratio for each constituent basic block of a malware variant within each malware family is computed. To avoid shared basic blocks in-between different malware families becoming a member of critical malicious patterns, the frequency distribution ratio

value is penalised for its occurrence in other malware families. The proposed classification method is based on two attributes, which are defined for every constituent basic block of a new malware variant, as presented in our recently published work [20]. Firstly, each basic block is scored based on its distribution frequency ratio in each malware family, and if it did not exist previously, it is ignored. Secondly, the distribution frequency of each basic block based on its occurrence in a number of different malware variants in each malware family is computed; according to both of these attributes for all of the constituent basic blocks of a malware variant, it is classified into a malware family.

In Section II, the related work and background on different malicious pattern matching techniques are reviewed. In Section III, the newly proposed formal method of malware classification, the applied methodology, and the implementation process is described. In Section IV, a performance evaluation of the experimental method is given. Finally in Section V, the newly proposed approach for malware classification is discussed and summarised.

II. Related Work and Background

Two main approaches that are utilised by malware researchers to analyse and detect malicious software are categorised as static and dynamic. In the following, the existing studies which have applied similar techniques to the newly proposed methodology have been reviewed, in order to set the background. In static analysis approach, different features of malicious software, and pattern matching techniques are studied to classify the enormous number of malware variants getting submitted to honeypots. N-gram frequency distribution is a well-known approach for extracting features and developing training dataset for classification of malware variants. An n-gram is the occurrence of a sequence of *n* characters in a string of assembly statement [2], raw bytes [10], [14], [15], and so on. N-gram computation as a pattern matching technique, and its application in malware analysis, was first introduced by authors in [7], [13], to identify boot sector viruses automatically by applying artificial neural networks. Authors in [15], presented a heuristic method to specify a suitable value for *n*. They collected a dataset of 1,971 benign executables, and 1,942 malicious software samples, and converted them to hexadecimal representation in *ASCII* format. They generated total of 255,904,403 distinct n-gram sequences by extracting sequences of four-bytes, and combining them into a single term. They computed the Information Gain (IG) [23], for each n-gram, selected top 500 n-grams, and applied different machine learning algorithms to detect and classify malware variants. They identified the boosted J48 algorithm as the best detector. Although they found good detection performance, with areas under the Receiver Operating Characteristic (ROC) curve of around 0.9, they did not consider metamorphic obfuscations and polymorphism.

The authors of [17], have followed the same methodology as the work presented in [15], differing only in producing the n-gram sequences. They generated the n-gram sequences as two-byte sequence which do not overlap, while the n-gram sequences generated by [15] may overlap and are in four-byte format. Comparatively, the authors in [15] obtained better performance, and developed a more comprehensive dataset collection. Both of the studies had difficulty in distinguishing between benign and malicious software, because of in-common behaviours such as editing the registry, or mass-mailing. Authors in [16], applied n-perm as variation of n-gram to ease some of the obfuscation techniques used in malware. N-perm is defined as a variation of n-gram, and it corresponds to every possible permutation of sequences of *n* characters, which are assigned as n-grams. The authors assumed that the order of characters in an n-perm sequence are independent of the n-gram sequence, but identical for the matching measurement. The results of their study show that applying exact matching of n-perm sequences, leads to better identification, by handling the presence of metamorphic techniques, such as code reordering, dead-code insertion, and instruction substitutions. The authors in [4], applied statistical analysis of opcode distribution in malicious software and benign software, as a dataset of in total 20 executable files. They disassembled all of the files by IDA Pro [1], and their opcode statistics were computed. The results showed the top five opcodes appearing in both the malicious and non-malicious software are: *mov*, *push*, *call*, *pop*, and *cmp*. Their preliminary study shows that less frequent shared opcodes, between the malicious and non-malicious software are better criteria to distinguish one from the other.

The authors of [24], extracted function features from disassembled malware variants. The function features include the count of API calls, the count of referenced strings, and the count of basic blocks. They did not consider the code, but only the counts, and applied Supported Vector Machines (SVMs), to select functions of interest, which are not shared in malware variants in different malware families. Therefore, the vectors with large difference from the average are extracted, and the quantity of the vectors for each malware variant is chosen based on the number of vectors "user" inputs, divided by the total number of functions in a malware. The classification process is performed by calculating the same values for the new malware variant, and comparing every function in the new malware variant with the functions in the feature set. However, the proposed approach has some flaws, such as the following: it requires the user to input the number of function features for each malware family to be involved in the classification, and the function features for each malware family are defined regardless to the distribution frequency ratio of the number of malware variants that exist in every malware family; also it does not provide much of the functionality relatedness between different malware families, for the malware analyst.

The authors in [5], presented a recent work on analysing PE files. They traced and captured API calls made by the malicious file, and extracted API call sequences from it. For the purpose of classification, they further generated n-grams from combination of the API call sequences, and calculated the Odds Ratio for each n-gram. Odds ratio, is a statistical calculation of the probability of presence or absence of a property. Finally, they used machine learning algorithms to classify malware and benign files. They show a detection accuracy of 98.5% as the outcome. However, the authors did not show a proper simulation outcome, and did not explain how the leading n features of the training dataset are selected. Also,

[1] I. Guilfanov, IDAPro, An Advanced Interactive Multi-processor Disassembler, Data Rescue. Available: http://www.datarescue.com

their dataset presented in the evaluation is rather too small to judge the results. The authors of [21], dynamically analysed malicious software through an emulator by observing its API calls at its run time. They extracted the critical API calls, which are the most significant to a specific family of malware variants, referred to as principal component analysis (PCA) features. The frequency of the observed significant API calls is used to create one signature for every malware family. The results of this study shows that considering critical API calls as a signature towards malware identification and filtration is effective. However, the authors generated signatures based on a limited number of malware variants, and due to changes in malware design, which evolve rapidly with time, ultimately new malware variants unrecognisable from the original form.

The authors in [9], applied fine-grained taint analysis on Central Processing Unit (CPU) instructions, and Direct Memory Access (DMA) operations, in order to find malicious software, which attempts to access information on the host system. They traced all the sensitive information and instructions through instrumentation, and as close as possible to the hardware, in order to produce taint graphs. Using the taint graphs, and based on predefined policies, the authors could identify and detect malicious software. They evaluated the effectiveness of their approach based on 42 malware samples and 56 benign software samples, and found very few false positives, and zero false negatives. However, the evaluation dataset is very small, and malware authors can easily obfuscate the malware to evade taint analysis. Also, implementation of this approach is very expensive. The authors of [3], presented a recent artefact, VirusBattle, which is a malware analysis web-service. VirusBattle reasons about malware variants in different levels of abstraction including the code, the statically analysed shared semantics, referred to as juice [1], amongst different variants, and the sequence of events that a malware sample undergoes during its execution, control flow graph (CFG) related code, to map the similarities and interrelationships. Juice transforms code semantics computed over a x86 disassembly by generalising the register names and literal constants, and computing the algebraic constraints between the numerical variables. Therefore, semantically similar code fragments can be identified by comparing their hash values. VirusBattle provides an automated PE unpacking web-service as well, which is a generic unpacker [2], and a publicly available web-service. In this paper, the unpacker provided by the VirusBattle SDK has been used to unpack the malicious samples, as well as the juice, which is the generalised presentation of the semantics to avoid code obfuscations.

III. Malware Classification

The classification of a new malware variant into one of the malware families is performed based on the the identified critical malicious patterns, as presented in a previous study [20]. The classification methodology is based on the Term Frequency Distribution (TFD) value, which is the key to evaluate the criticality of a basic block, and the count of malware variants in each malware family which contain that basic block. These fundamentals are defined based on our experimental evaluations, as described in the following sections.

A. Formalisation

Considering each malware family, f, as a set of malware variants, and each malware variant be a set of multiple basic blocks. The malware families are disjoint, in other words each malware variant can only be a member of one malware family. Also, considering that a new malware variant to be classified is a multiset of basic blocks, v, it is assumed that v_i, becomes a member of every existing f in the dataset, one by one. However, v_i, is the ith distinct member of a v which can occur in any malware family and multiple times.

Definition 1: Let f_k be the kth distinct member of f, and let $\rho_{v_i,k}$ be the total count of malware variants belonging to f_k, which contain v_i, and let δ_k be the total number of malware variants in f_k. Therefore, the distribution frequency of v_i in f_k, in relation to the number of malware variants containing it is defined as MDF_{v_i}, and is referred to as the Malware Distribution Frequency of v_i, as shown in Equation 1.

$$MDF_{v_i} = \frac{\rho_{v_i,k}}{\delta_k} \tag{1}$$

The count of malware variants in every malware family in which a specific basic block occurs, influences the criticality of that basic block to be a potential candidate of a critical malicious pattern for that malware family. As it affects the criticality of a basic block more when it is the constituent basic block of a greater number of malware variants of a malware family, it is computed by MDF_{v_i}. For example, if basic block v_i occurs in f_k in 2 malware variants, it should be less effective comparatively, than in f_j in which 5 malware variants contain it. Furthermore, to compare the effectiveness of MDF_{v_i} on the classification, a new malware variant is classified by scoring its constituent basic blocks according to its TFD value, as shown in [20], as well as $FamClassifier$ as it is defined below.

Definition 2: The classification algorithm is defined as multiplication of $TFD_{i,k}$ by MDF_{v_i}, as expressed in Equation 2.

$$FamClassifier_{i,k} = TFD_{i,k} \times MDF_{v_i} \tag{2}$$

B. Malware Classification Methodology

A new malware variant is classified into its related malware family based on scoring of each of its v_i, according to the value of $FamClassifier$, or TFD, and the malware family which gets the highest score becomes the classification family. The highest score is allocated to the score of a malware family which gets the highest number of top TFD value or top $Famclassifier$ value for most of the constituent basic blocks of the new malware variant. $FamClassifier$ and TFD values are computed for every distinguished basic block, v_i, of the new malware variant. Processed malware variants are either classified, or they remain unclassified based on the current state of the dataset. Processed malware variants are the unpacked new malware variants, and in the readable format by the classifier. The readable format for the classifier is a multiset of

[2]V. Notani, A. Lakhotia, and et al. VirusBattle SDK-Unpacker. Available: https://bitbucket.org/srl/virusbattle-sdk/wiki/Credits%20for%20VirusBattle

basic blocks for each malware variant, which are hashed. If a malware variants is classified successfully, then the state of the current dataset is updated. After each update of the dataset, the $TFD_{i,k}$, and MDF_{v_i} value for the constituent basic blocks of the malware variants in the unclassified malware samples are updated and checked by the classifier for possible further matches. This update and further similarity measurement is performed due to the fact that every time a malware variant replicates, it carries some new pieces of code. However, it is not completely a new firmware, and still carries shared code from its previous version. So it is possible to identify it based on the shared code and classify it into its related malware family, noticing that some new code is also introduced into the corresponding malware family which may match a variant in the unclassified malware samples. The initial dataset is developed with a limited number of malware families, 23, but obviously there are more malware families in the wild, and from time to time a new malware family is introduced by the malware authors. Therefore, there is a high possibility that the unclassified malware variants do not belong to any malware family existing in the dataset. Thus, another module checks if a number of unclassified malware variants share a certain amount of malicious code (critical malicious pattern) together, then groups them together and introduces a new possible malware family into the dataset. This is the cycle of classification of the new malware variants.

C. Experimental Set-up

In order to make the experiment as accurate as possible, the following parameters are applied. Every malware variant is classified into a malware family based on the similarity measurement of its constituent basic blocks in comparison to the critical malicious patterns of different malware families which exist in the dataset. As described earlier, the most critical malicious patterns in every malware family, the authors have relied on the shared basic blocks among different malware variants, which potentially occur in one specific family at most [20]. To do so, $TFD_{i,k}$ and $FamClassifier_{i,k}$ need to be computed for all the constituent basic blocks of the new malware variant to make it comparable with the critical malicious patterns. If a basic block existed previously in the dataset, it will be assigned the $TFD_{i,k}$, and if it has not occurred before, it will be assigned a $null$ value and will not be involved in the classification. In other words, a new malware variant will be classified based on its constituent basic blocks, which have already occurred in one or more malware variants in the dataset, and if none of its basic blocks occurred previously it is considered to be unclassified. To achieve this, an unlabelled malware family is created, and the new malware variants are added to this family one by one in order to include the new variants' constituent basic blocks in the dataset and compute the required values. Therefore, initially for all the constituent basic blocks of the new malware variant, just as for any other malware variants existing in the dataset, the required values such as $TFR_{i,k}$, $TFD_{i,k}$, MDF_{v_i}, and $FamClassifier_{i,k}$ are computed, unless it has not occurred previously. Two main queries which make the classification possible are as listed below:

1) *classify fc [path] scoring;*
2) *classify tfd [path] scoring;*

In the first query, fc stands for $FamClassifier$, and it is computed for every v_i of the new malware variants; *path* is the path of the new malware variants to be classified, and *scoring* lists the corresponding malware families in which v_i has occurred and v_i is its critical malicious pattern in comparison with other malware families. Therefore, for every occurrence of v_i as a critical malicious pattern of f_k, f_k receives a positive score and the malware family with the highest score will be appointed as the best match for classification. The malware family with the highest score signifies that it shares more critical malicious patterns with the new malware. New malware variants to be classified can be added to the dataset individually or in a batch form; however, they are classified one by one, and the status of the dataset is updated after every addition process, regardless of being added to the classified or unclassified category. Query 2, follows the same approach as query 1 in terms of scoring, but classification is performed based on the $TFD_{i,k}$ value. This means that the malware distribution frequency, MDF_{v_i} has not influenced the classification.

D. Unclassified Malware

Unclassified malware, refers to the malware variant with which the classifier is not able to associate a malware family. The reason for not being able to classify a malware variant based on the current implementation of the classification method is because no shared basic blocks exist between the new malware variant and the dataset. Although not yet implemented, the proposed approach, as described earlier, is to store all the unclassified malware variants in one malware family and label it as unclassified; after every successful classification of new malware variants and by growing the size of the dataset, they will be again checked for a possible match. Also, by measuring the similarity of unclassified malware variants with one another, if a number of malware variants in the unclassified malware group share a certain number of basic blocks, they will be associated together to create a new malware family. Allowing the introduction of new malware families and growing the dataset can be supervised by administrator intervention for more accurate classification.

IV. Performance Evaluation

Performance analysis of the proposed method includes evaluation of the accuracy of the experimental results, and the running time of the malware classification methods. In the following, the effectiveness and efficiency of the proposed method is inspected, as well as the degree of classification correctness by performing cross validation on the dataset. The classification correctness and effectiveness depends upon correct malware family assignment of the malware variants when developing the training dataset; in other words, initially labelling the malware variants according to antivirus scanner results, as well as the newly defined formal methods. To assess the classification results, we count the number of false matches (F) and true (T) matches, and compute the accuracy of the results accordingly. The accuracy of known and unknown malware variants is measured as defined in Equation 3:

$$Accuracy = \frac{T}{(T+F)} \tag{3}$$

We expect the accuracy for each malware family classification to be close to 1, which indicates how closely the test determines true matches.

A. Effectiveness and Efficiency

Effectiveness and efficiency refer to the responsiveness and running time of the classification algorithm, given the consumed resources. The running time to classify each malware variant is measured by means of the embedded method $timeCommand()$, and it can be obtained when classification scoring is presented for each malware family. For instance, the elapsed time to classify a malware variant by $FamClassifier$ on average lies between 26 seconds and 42 seconds, and by TFD lies betweem 0.09 seconds and 0.12 seconds. Clearly there is a huge gap between the two, and one of the reasons for this is the implementation design. The values of $tfd_{i,k}$ and MDF_{v_i} for all the basic blocks in the dataset, are updated after every malware classification. However, $famClassifier$ takes a longer time to classify as it is required to loop through all the malware variants in the dataset and compute the occurrences of every MDF_{v_i} in each malware family, which is a resource consuming and time consuming process. To check the effectiveness of the classifiers in classifying known malware variants, we use the same test set as the training dataset itself. The classifiers are able to classify all malware variants, but there are occurrences of false classifications by both of the classifiers. $FamClassifier$ shows better classification results on known malware variants than TFD, with 2% improvement. More details on each malware family classification and corresponding accuracy are given in the next part, Section IV-B. To determine whether $FamClassifier$ or TFD shows a better response in terms of unknown malware variants classification, *K-fold* cross validation on the dataset is employed. In other words, the effectiveness of critical malicious patterns and malware distribution frequency of each basic block, on the classification outcome is assessed. To test the classification algorithms, each malware family is randomly partitioned into 5 subsets of approximately the same number of variants, and similar file size. Testing is repeated 5 times, and every time one subset of the dataset is used as the test set, and the other 4 subsets are combined together to form the training dataset. Furthermore, training dataset is used to calibrate the test and validate the effectiveness of the classification result. The result shows that both of the classifiers are able to classify all the malware variants in the test datasets. Contrary to the results for known malware variants classification, TFD outperforms $FamClassifer$ in terms of accurate classification and also the duration it takes to classify; details on the classification results are given in Section IV-B.

B. Correctness of Classification Algorithms

By correctness of classification algorithms, we refer to the accuracy of classifying known and unknown malware variants into malware families. In more technical terms, the rate of false matches and that of true matches of the classification. For the TFD classifier the accuracy rate is 93.77%, which implies the misclassified rate of known malware variants based on the most critical malicious patterns of every malware family is 6.23%. This misclassification rate can be caused by wrong initial preprocessing and labelling of the malware samples. However, the $FamClassifier$ classification method shows better results in terms of known malware classification, although the duration of its execution time highly exceeds that of the TFD classifier as discussed previously. More than 95% of known malware variants are classified correctly, which means MDF_{v_i} has 1.23% positive effect on the accuracy of known malware classification. Unexpected or unknown malware variants classification results are in the reverse order, and the TFD classifier outperforms the $FamClassifier$ by approximately 4%. The TFD classifier has an accuracy rate of 83.44% and the $FamClassifier$ accuracy rate is 79.73%. This reversed accuracy rate of classification results between known and unknown malware variants can be explained as the count of malware variants containing a specific basic block in a malware family does not particularly affect the criticality of that basic block as being a potential representative of critical a malicious pattern for unknown malware variants classification. Possible reasons for this will be discussed in more detail in Section V. Looking into the accuracy rate of different malware families classification with regard to known malware variants, *Unruy* family holds the lowest accuracy rate by either of the classifiers, and with regard to unknown malware variants, *ATRAPS* family has the lowest accuracy rate for both of the classifiers; however, no specific malware families shows the highest accuracy classification rate for either of the classifiers.

The result of known malware variants classified by the TFD classifier shows 8 of 23 malware families are classified 100% accurately, 7 of 23 malware families are classified with average accuracy rate of 95%, 7 of 23 malware families are classified with average accuracy rate of above 85%, and 1 last malware family is classified with approximate accuracy rate of 72%. The results of known malware variants classified by the $FamClassifier$ shows 6 of 23 malware families are classified 100% accurately, 10 of 23 malware families are classified with average 93% accuracy rate, and the remaining 6 malware families are classified with average accuracy rate of 85%, and 1 last malware family is classified by lowest classification accuracy of 75%. Furthermore, the results of unknown malware variants classified by the TFD classifier shows that only 1 malware family is classified 100% accurately which is *Cosmu* family, 4 of 23 malware families are classified with average accuracy rate of 94%, 8 of 23 are classified with average accuracy rate of 85%, 4 of 23 malware families are classified with average accuracy rate of 75%, 4 of 23 malware families are classified with average accuracy rate of 66%, and the remaining 2 malware families are classified with average accuracy rate of 50%; and result of unknown malware variants classified by t$FamClassifier$ shows no malware family is classified 100% accurately, 3 of 23 malware families are classified with average accuracy rate of 96%, 5 of 23 malware families are classified with average accuracy rate of 85%, 6 of 23 malware families are classified with average accuracy rate of 76%, 4 of 23 malware families are classified with average accuracy rate of 66%, 2 of 23 malware families are classified with average accuracy rate of 54%, and 3 of 23 malware families are classified with average accuracy rate of 44%.

V. Discussion and Conclusion

To understand the classification accuracy rate outcome, we looked further into one of the well classified malware

families, *MyDoom*, and one of the less accurate classified malware families, *Unruy* were investigated further. The top 10 values of TFD on basic blocks of both of the families were queried. There is an obvious difference between the distribution frequency and count of basic blocks of the named malware families. This indicates that the basic blocks in the *Unruy* family are less in-common in comparison to *MyDoom*, and provides us with a good understanding of the functional structure of *Unruy* family. Comparing more number of the malware families in the dataset in the same way reveals a shortcoming in the preprocessing phase, which is that the closeness of the replication versions of the malware variants in each malware family were not observed. Malware variants from the same malware family can be very different from one another as the replication timeline between the versions increases. However, this can be addressed by adding a threshold on the accepted TFD value for v_i to become a critical malicious pattern and affect the classification, which is an issue for the implementation, but not the concept. Also, it was observed that malware families such as *Spyware, and Kryptik*, which share a certain amount of functionality with each other can cause misclassification. For example both of the malware families tend to install on the host machine and obtain information about the user. Therefore, these malware families share very similar components and as the present classification method is based on the basic blocks which represent the functionality of the malware families, it can produce false matches. This observation suggests that the malware families defined in the training dataset should be more coarse grained in the functionality of different malware families, when preparing the training dataset. Another reason may be the labelling by the antivirus scanners that we used to pre-classify the training dataset, since several malware variants are tagged by different antivirus scanners as belonging to multiple malware families, and we chose the most frequent one. Therefore, this could also be the result of incorrect initial labelling.

As explained previously, the TFD classifier shows a better classification accuracy rate, and the time it takes to classify new malware variants is insignificant in comparison to $FamClassifier$. This is because the *update* module refreshes the TFD_i, k and MDF_{v_i} after every single malware classification. $FamClassifier$ takes much longer because the computation of MDF_{v_i} at run time, which involves the execution of multiple loops. Multiple loops are implemented as it is measured for every basic block in every malware family, and needs to be checked for its occurrence in every malware variant. However, considering the time and accuracy of both of the classifiers, the TFD classifier is preferred and it is proposed to involve more details of the malicious software to be included in the critical malicious pattern to increase the accuracy, such as the strings, the information contained in the import-export tables of the malware, the file type, etc. to potentially improve the accuracy rate. Nonetheless, previous studies have shown that combining different features extracted from malware variants for classification purposes does not necessarily produce better results, as it can produce noise in the outcome [22], but the present approach has not been tried before and could lead to a different outcome. Nevertheless, it is contended that the proposed approach in the present research is simple, accurate, and effective. It can be applied to other features of the malicious software, by extracting the feature of interest, computing its hash value, and applying the TFD algorithm to obtain the pattern for each malware family. Thus, the present methodology can be used to retrieve critical information about each malware family, which is essential to understand malicious software, while also generating an effective general pattern for the numerous malware variants of each malware family. Based on these results, there is no specific relationship between the number of malware variants in each malware family and the classification accuracy rate, which is explicable since the ratio of frequencies of different features is computed with respect to the size of the training dataset. The evaluation of the present formalisation is relatively straightforward since the formalisation is based on the statistics. The counts, and the implementation outcomes have been checked manually and found to be accurate. The selection of the present malware families and the basic blocks in the listings and examples are in principle arbitrary.

The main drawback of our approach is the size of the dataset. Clearly, a greater number of malware variants and malware families will provide more details on each malware family and lead to more precise critical malicious patterns. Another important factor is the accuracy of labelling the malware variants by anti-virus scanners, to develop the training dataset, as every wrong pre-classification of malware variants will lead to a misclassified outcome in the evaluation phase. Nonetheless, in this research, the proposed concept is being proofed and these factors are relatively trivial. In this paper, the statistical approach recently published in [20] has been followed, which discovers and generates critical malicious patterns in malware families, by which new malware variants are classified accordingly. Critical malicious patterns are defined as the most frequent basic blocks in one malware family, while relatively less frequent in other malware families. Moreover, the critical malicious pattern of every malware family has been used as a criterion for new malware variants classification. Classification is performed by two methods, one is the TFD classifier which computes the similarity measurement of the new malware variant with the malware families that exist in the training dataset, based on the critical malicious pattern; $FamClassifier$ is the other classifier which classifies malware variants based on the critical malicious pattern, as well as malware frequency distribution of every distinct basic block in a malware family. The effectiveness and accuracy rate of both of the classifiers has been evaluated and the TFD classifier is preferred, as it outperforms $FamClassifier$ in time and classification of unknown malware variants. In addition, it is concluded that there is no specific association between the number of malware variants in the malware families, and the classification accuracy rate that has been observed, and this is explicable by the definition of the formalisation, which is based on the frequency ratio of the basic blocks occurrences. The possible inclusion of addition malware variant features as part of the classification criteria is proposed to increase the classification accuracy rate, as part of the future work. Also, the possible mislabelling of the malware variants in the preprocessing phase of this research by anti-virus scanners, has been observed as one of the impact factors of false matches by the classifiers. Finally, an increase in the number of malware variants in the training dataset is recommended, in order to aid wider evaluation of the classifiers in terms of responsiveness and correctness.

REFERENCES

[1] A. Lakhotia, M. D. Preda and R. Giacobazzi, *Fast Location Of Similar Code Fragments Using Semantic 'Juice'*, Proceedings of the 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop: ACM, 5, 2013.

[2] A. Walenstein, M. Venable, M. Hayes, C. Thompson and A. Lakhotia, *Exploiting Similarity Between Variants To Defeat Malware*, In Proceedings of BlackHat, 2007.

[3] C. Miles, A. Lakhotia, C. LeDoux, A. Newsom and V. Notani, *Virusbattle: State-Of-The-Art Malware Analysis For Better Cyber Threat Intelligence*, Resilient Control Systems (ISRCS), 2014 7th International Symposium on: IEEE, 1–6, 2014.

[4] D. Bilar, *Opcodes As Predictor For Malware*: International Journal of Electronic Security and Digital Forensics, 156–168, 2007.

[5] D. Uppal, R. Sinha, V. Mehra and V. Jain, Vinesh, *Malware Detection And Classification Based On Extraction Of Api Sequences*, Advances in Computing, Communications and Informatics (ICACCI), 2014 International Conference on: IEEE, 2337–2342, 2014.

[6] F. Cohen, *Computer Viruses: Theory And Experiments*: Computers & Security 6, 22–35, 1987.

[7] G. Tesauro, J. F. Kephart and G. B. Sorkin, *Neural Networks for Computer Virus Recognition*: IEEE Expert, 5–6, 1996.

[8] G. Wagener, R. State and A. Dulaunoy, *Malware Behaviour Analysis*, Journal in Computer Virology: Springer, 279–287, 2008.

[9] H. Yin, D. Song, M. Egele, C. Kruegel and E. Kirda, *Panorama: Capturing System-Wide Information Flow For Malware Detection And Analysis*, Proceedings of the 14th ACM conference on Computer and communications security: ACM, 116–127, 2007.

[10] I. Santos, C. Laorden, P. G. Bringas, *Collective Classification For Unknown Malware Detection*, In Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference: CEAS, 2011.

[11] I. You and K. Yim *Malware Obfuscation Techniques: A Brief Survey*, Broadband, Wireless Computing, Communication and Applications (BWCCA): International Conference, 297-300, 2010.

[12] J. Giffin and A. Srivastava, *Attribution of Malicious Behavior*, Information Systems Security: Springer, 28–47, 2010.

[13] J. O. Kephart, G. B. Sorkin, W. C. Arnold, D. M. Chess,G. J. Tesauro, S. R. White, *Biologically Inspired Defences Against Computer Viruses*, Proceedings of the 14th International Joint Conference on Artificial Intelligence: IJCAI, 985–996, 1995.

[14] L. Goldberg, G. Leslie Ann, Ph. W. Paul, A. S. Cynthia and B. Gregory. *Constructing Computer Virus Phylogenies*, 1996.

[15] J. Z. Kolter and M. A. Maloof, *Learning To Detect And Classify Malicious Executables In The Wild*, Journal of Machine Learning Research: JMLR, 2721–2744, 2006.

[16] Md. E. Karim, A. Walenstein, A. Lakhotia and L. Parida, *Malware Phylogeny Generation Using Permutations Of Code*: Journal in Computer Virology, 13–23, 2005.

[17] M. G. Schultz, E. Eskin, E. Zadok and S. J. Stolfo, *Data Mining Methods For Detection Of New Malicious Executables*, Security and Privacy, IEEE Symposium on: IEEE, 00–38, 2001.

[18] O. Sukwong, H. S. Kim and J. C. Hoe, *Commercial antivirus software effectiveness: an empirical study*: IEEE, 63–70, 2011.

[19] P. Barford and V. Yegneswaran, *An Inside Look At Botnets*, Malware Detection: Springer, 171–191, 2007.

[20] V. Ghanaei, C. S. Iliopoulos, and R. E. Overill, *A Statistical Approach for Discovering Critical Malicious Patterns in Malware Families*, The Seventh International Conferences on Pervasive Patterns and Applications, (Patterns 2015): IARIA, 2015.

[21] V. P. Nair, H. Jain,Y. K. Golecha, M. S. Gaur and V. Laxmi, *MEDUSA: Metamorphic Malware Dynamic Analysis Using Signature From API*, Proceedings of the 3rd international conference on Security of information and networks: ACM, 263–269, 2010.

[22] W. M. Khoo, A. Mycroft,R. Anderson *Rendezvous: A Search Engine For Binary Code*, Proceedings of the 10th Working Conference on Mining Software Repositories: IEEE Press, 329–338, 2013.

[23] Y. Yang and J. O. Pedersen, *A Comparative Study On Feature Selection In Text Categorization*, In Proceedings of ICML-97, 14th International Conference on Machine Learning: US: Morgan Kaufmann Publishers, 412–420, 1997.

[24] Y. Zhong, H. Yamaki and H. Takakura, *A Malware Classification Method Based On Similarity Of Function Structure*, Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on: IEEE, 256–261, 2012.